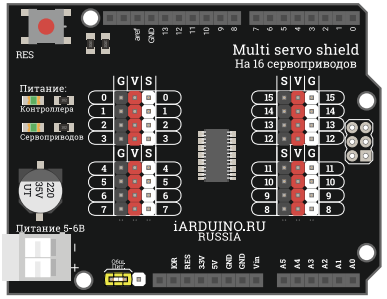
Multi Servo Shield (на 16 сервоприводов)



Общие сведения:

Multi Servo Shield на 16 сервоприводов - используется для управления сервоприводами по шине I2C.

Спецификация:

- Напряжение питания прикладываемое к выходам: 5 ... 6 В.
- Ток нагрузки на выходах: до 25 мА.
- Частота тактирования: 25 МГц внутренний генератор (±3%).
- Количество каналов ШИМ: 16 шт.
- Разрешение ШИМ: 12 бит 4096 тактов (от 0 до 100%).
- Выходная частота ШИМ: 24 ... 1526 Гц (для сервоприводов устанавливается в 50 Гц)
- Рабочая частота шины I2C: ... 100кГц, 400кГц, 1МГц.
- Рабочая температура: -40 ... 85 °C.

Подключение:

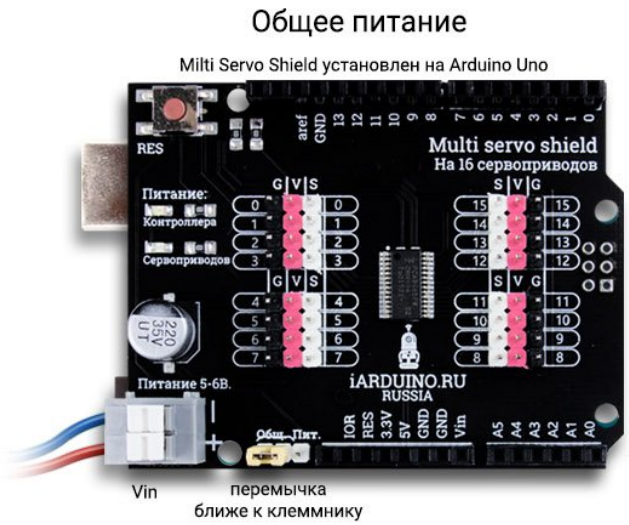
Servo Shield устанавливается на Arduino UNO и использует только два аппаратных вывода шины I2C. Адрес Servo Shield на шине 0x40.

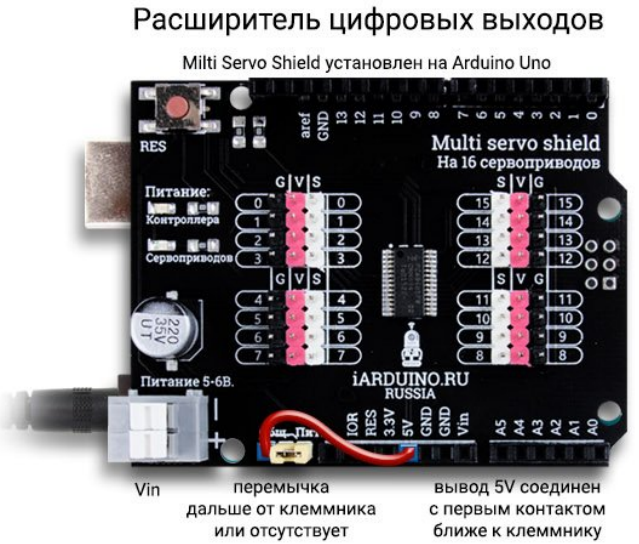
Питание:

- Если Servo Shield используется для управления сервоприводами, то выходное напряжение берется ТОЛЬКО от внешнего источника питания 5 - 6 В, который подключается к клеммнику Vin.
- Если Servo Shield используется как расширитель цифровых выходов Arduino, то внешнее питание на Servo Shield не подаётся.

Рядом с клеммником питания расположены три контакта с перемычкой.

- Если перемычка замыкает контакты расположенные ближе к клеммнику «Общ. Пит.», то питание с Servo Shield подаётся на вход Arduino «Vin» (общее питание берется с Servo Shield). Не требуется отдельного питания Arduino.
- Если перемычка расположена на контактах дальше от клеммника, или отсутствует, то питание должно подаваться на Arduino, а на Servo Shield подаётся только в том случае, если он используется для управления сервоприводами (раздельное питание).





Подробнее о Servo Shield:

Servo Shield построен на базе чипа PCA9685, все функции которого, реализованы в библиотеке iarduino_MultiServo.

Данный Shield позиционируется как устройство управления сервоприводами, но его можно использовать и в качестве расширителя цифровых выходов Arduino (не входов, а только выходов) с возможностью вывода сигналов ШИМ, в таком случае внешнее питание на клеммник Vin не подаётся.

Для работы с Servo Shield предлагаем воспользоваться библиотекой iarduino_MultiServo, в которой реализован весь функционал чипа PCA9685.

Подробнее про установку библиотеки читайте в нашей инструкции.

Примеры:

Управление сервоприводами: SG90, MG90, MG996R, Futaba S3003:

```
#include <iarduino_MultiServo.h>        //  Подключаем библиотеку iarduino_MultiServo
iarduino_MultiServo MSS;                //  Объявляем объект MSS, для работы с библиотекой
void setup() {
// Указываем какой тип сервопривода подключен к выводам MultiServoShield
   MSS.servoSet(0, SERVO_SG90);         //  Сервопривод SG90 подключён к выводу № 0 MultiServoShield.
   MSS.servoSet(1, SERVO_SG90);         //  Сервопривод SG90 подключён к выводу № 1 MultiServoShield.
   MSS.servoSet(2, SERVO_SG90);         //  Сервопривод SG90 подключён к выводу № 2 MultiServoShield.
   MSS.servoSet(3, SERVO_MG90);         //  Сервопривод MG90 подключён к выводу № 3 MultiServoShield.
   MSS.servoSet(4, SERVO_MG996R);       //  Сервопривод MG996R подключён к выводу № 4 MultiServoShield.
   MSS.servoSet(5, SERVO_FutabaS3003);  //  Сервопривод Futaba S3003 подключён к выводу № 5 MultiServoShield.
// MSS.servoSet(SERVO_ALL, SERVO_SG90); //  На любом выводе используются только сервоприводы SG90
   MSS.begin();                         //  Инициируем работу с MultiServoShield
}
void loop(){
// Управляем сервоприводами
   MSS.servoWrite(3, 50);               //  Повернуть сервопривод, подключённый к 3 выводу, на 50°
   delay(1000);                         //  Ждём 1 сек.
   MSS.servoWrite(5, 120);              //  Повернуть сервопривод, подключённый к 5 выводу, на 120°
   delay(1000);                         //  Ждём 1 сек.
   MSS.servoWrite(SERVO_ALL, 180);      //  Повернуть все сервоприводы на 180°
```

```
   delay(1000);                          //  Ждём 1 сек.
   MSS.servoWrite(SERVO_ALL, 0);         //  Повернуть все сервоприводы на 0°
   delay(1000);                          //  Ждём 1 сек.
}
```

Управление любыми сервоприводами:

```
#include <iarduino_MultiServo.h>         //  Подключаем библиотеку iarduino_MultiServo
iarduino_MultiServo MSS;                 //  Объявляем объект MSS, для работы с библиотекой
void setup() {
// Указываем какой тип сервопривода подключен к выводам MultiServoShield
   MSS.servoSet(0, 180, 130, 470);       //  К выводу 0, подключен сервопривод с максимальным углом поворота 180°, ШИМ для 0° =
   MSS.servoSet(1, 120, 100, 500);       //  К выводу 1, подключен сервопривод с максимальным углом поворота 120°, ШИМ для 0° =
// MSS.servoSet(SERVO_ALL, 90, 50, 300);//  На любом выводе используются сервоприводы с максимальным углом поворота 90°, ШИМ д
   MSS.begin();                          //  Инициируем работу с MultiServoShield
}
void loop(){
// Управляем сервоприводами как и в предыдущем примере
}
```

Управление выходами Arduino: (расширитель выходов)

```
#include <iarduino_MultiServo.h>         //  Подключаем библиотеку iarduino_MultiServo
iarduino_MultiServo MSS;                 //  Объявляем объект MSS, для работы с библиотекой
void setup() {
   MSS.begin(0x40, 1000);                //  Инициируем работу с MultiServoShield
                                         //  указывая адрес MultiServoShield на шине I2C: 0x40 (по умолчанию 0x40)
                                         //  и частоту сигнала ШИМ: 1000 Гц (по умолчанию 50 Гц), можно указать от 1 до 1526 Гц
}
void loop(){
   MSS.digitalWrite(5, LOW);             //  Устанавливаем на 5 выводе     низкий  уровень (уровень логического «0»)
```

```
  MSS.digitalWrite(6, HIGH);           //  Устанавливаем на 6 выводе     высокий уровень (уровень логической  «1»)
  MSS.digitalWrite(SERVO_ALL, LOW);    //  Устанавливаем на всех выводах низкий  уровень (уровень логического «0»)
  MSS.digitalWrite(SERVO_ALL, HIGH);   //  Устанавливаем на всех выводах высокий уровень (уровень логической  «1»)
  MSS.analogWrite(0, 1023);            //  Устанавливаем на 0 выводе     сигнал ШИМ с коэффициентом заполнения 1023 (25%), до
  MSS.analogWrite(1, 2047);            //  Устанавливаем на 1 выводе     сигнал ШИМ с коэффициентом заполнения 2047 (50%), до
  MSS.analogWrite(SERVO_ALL, 3071);    //  Устанавливаем на всех выводах сигнал ШИМ с коэффициентом заполнения 3071 (75%), до
  MSS.analogWrite(10, 4095);           //  Устанавливаем на 10 выводе    сигнал ШИМ с коэффициентом заполнения 4095 (100%), д
  MSS.analogWrite(2, 2047, 1023);      //  Устанавливаем на 2 выводе     сигнал ШИМ с коэффициентом заполнения 2047 (50%), до
                                       //  и фазовым сдвигом 1023 (25%), допускаются значения от 0 до 4095.
                                       //  Фазовый сдвиг - необязательный параметр указывается от 0 до 4095 (по умолчанию 0)
                                       //  устанавливает задержку сигнала ШИМ по отношению к другим выводам, но не влияет на
  MSS.analogRead(5);                   //  Читаем ранее установленный коэффициент заполнения ШИМ на выводе 5
                                       //  если на выводе был установлен низкий  уровень (LOW),  то функция вернёт 0
                                       //  если на выводе был установлен высокий уровень (HIGH), то функция вернёт 4096
                                       //  в нашем случае, функция вернёт 3071
```

Описание основных функций библиотеки:

Данная библиотека может использовать как аппаратную, так и программную реализацию шины I2C.
О том как выбрать тип шины I2C рассказано в статье Wiki - расширенные возможности библиотек iarduino для шины I2C.

Подключение библиотеки:

```
#include <iarduino_MultiServo.h>       // Подключаем библиотеку.
iarduino_MultiServo MSS;               // Объявляем объект MSS, для работы с Multi Servo Shield.
```

Функция begin();

- Назначение: Инициализация работы с Multi Servo Shiel.
- Синтаксис: begin([АДРЕС [, ЧАСТОТА]]);
- Параметры:
  - АДРЕС - адрес Shield на шине I2C, указывается от 0 до 127 (по умолчанию 0x40).

    - ЧАСТОТА - Частота ШИМ, указывается в Гц от 1 до 1526 (по умолчанию 50 Гц).
- Возвращаемые значения: Нет.
- Примечание: Достаточно вызвать 1 раз в коде Setup, но допускаются и повторные вызовы, если требуется изменить частоту или адрес.
- Пример:

```
MSS.begin();                          // Инициируем работу с MultiServoShield, используя значения по умолчанию (адрес на шин
MSS.begin(0x41);                      // Инициируем работу с MultiServoShield, используя адрес на шине I2C = 0x41 и частоту
MSS.begin(0x40, 1000);                // Инициируем работу с MultiServoShield, используя адрес на шине I2C = 0x40 и частоту
```

Функция servoSet();

- Назначение: Установка параметров для каждого подключённого сервопривода.
- Синтаксис:
    - servoSet(№_ВЫХОДА , НАЗВАНИЕ); // Только для сервоприводов SG90, MG90, MG996R, Futaba S3003.
    - servoSet(№_ВЫХОДА , МАКС_УГОЛ , ШИМ_МИН , ШИМ_МАКС); // Для любых сервоприводов.
- Параметры:
    - №_ВЫХОДА - число, от 0 до 15. Если указать SERVO_ALL, то параметры применятся ко всем выходам.
    - НАЗВАНИЕ - одно из названий: SERVO_SG90, SERVO_MG90, SERVO_MG996R, SERVO_FutabaS3003.
    - МАКС_УГОЛ - число в градусах, от 1 до 360, указывающее максимально возможный угол сервопривода.
    - ШИМ_МИН - коэффициент заполнения ШИМ, от 0 до 4095, для угла 0°.
    - ШИМ_МАКС - коэффициент заполнения ШИМ, от 0 до 4095, для угла МАКС_УГОЛ.
- Возвращаемые значения: Нет.
- Примечание: Достаточно вызвать 1 раз, как до, так и после функции begin, но до функции servoWrite.
- Пример:

```
MSS.servoSet(SERVO_ALL, SERVO_SG90   ); // Указываем, что все подключённые сервоприводы являются SG90.
MSS.servoSet(14       , SERVO_MG996R ); // Указываем, что сервопривод на 14 выводе является MG996R.
MSS.servoSet(15       , 180, 200, 500); // Указываем, что сервопривод на 15 выводе имеет максимальный угол поворота 180°, углу
```

Функция servoWrite();

- Назначение: Поворот указанного сервопривода на заданный угол.
- Синтаксис: servoWrite(№_ВЫХОДА , УГОЛ);
- Параметры:
  - №_ВЫХОДА - число, от 0 до 15. Если указать SERVO_ALL, то повернутся все сервоприводы.
  - УГОЛ - число в градусах, от 0° до 360°, указывающее угол поворота для сервопривода.
- Возвращаемые значения: Нет.
- Примечание: Сервопривод не повернётся на угол, больше, чем указанный в функции servoSet().
- Пример:

```
MSS.servoWrite(SERVO_ALL, 100);         // Повернуть все сервоприводы на угол 100°.
MSS.servoWrite(14       , 120);         // Повернуть сервопривод, подключённый к 14 выходу, на угол 120°.
```

Функция analogWrite();

- Назначение: Установка сигнала ШИМ на указанном выходе, с указанным коэффициентом заполнения.
- Синтаксис: analogWrite(№_ВЫХОДА , ШИМ [, ФАЗОВЫЙ_СДВИГ]);
- Параметры:
  - №_ВЫХОДА - число, от 0 до 15. Если указать SERVO_ALL, то значения применятся ко всем выходам.
  - ШИМ - коэффициент заполнения ШИМ, лежит в пределах от 0 до 4095.
  - ФАЗОВЫЙ_СДВИГ - число, от 0 до 4095 (по умолчанию 0)
- Возвращаемые значения: Нет.
- Примечание:
  - Функция работает как одноимённая функция для обычных выходов ШИМ, но в диапазоне 0...4095.
  - Фазовый сдвиг, задерживает начало каждого импульса ШИМ по отношению к другим выводам.

```
MSS.analogWrite(SERVO_ALL, 3071);       // Устанавливаем на всех выходах сигнал ШИМ с коэффициентом заполнения 3071 (75%).
MSS.analogWrite(10,        4095);       // Устанавливаем на 10 выходе    сигнал ШИМ с коэффициентом заполнения 4095 (100%).
MSS.analogWrite(2,         2047, 1023); // Устанавливаем на 2 выходе     сигнал ШИМ с коэффициентом заполнения 2047 (50%) и фа
```

Функция digitalWrite();

- Назначение: Установка логического состояния LOW или HIGH на выходе.
- Синтаксис: digitalWrite(№_ВЫВОДА , СОСТОЯНИЕ);
- Параметры:
  - №_ВЫВОДА - число, от 0 до 15. Если указать SERVO_ALL, то состояние установится на всех выходах.
  - СОСТОЯНИЕ - одно из логических состояний LOW или HIGH.
- Возвращаемые значения: Нет.
- Примечание: Работает как одноимённая функция для обычных цифровых выходов.

```
MSS.digitalWrite(SERVO_ALL, LOW);       // Устанавливаем на всех выходах, уровень логического 0.
MSS.digitalWrite(0, HIGH);              // Устанавливаем на 0 выходе, уровень логической 1.
```

Функция analogRead();

- Назначение: Чтение ранее установленного коэффициента заполнения ШИМ.
- Синтаксис: analogRead(№_ВЫВОДА);
- Параметры:
  - №_ВЫВОДА - число, от 0 до 15, указывающее выход состояние которого требуется проверить.
- Возвращаемые значения:
  - Ранее установленный коэффициент заполнения ШИМ от 0 до 4095.
  - Если был установлен постоянный уровень LOW, то функция вернёт 0
  - Если был установлен постоянный уровень HIGH, то функция вернёт 4096.
  - Примечание: Работает как одноимённая функция для аналоговых входов, но в диапазоне 0...4096.
- Пример:

```
int i = MSS.analogRead( 5 );            // Присваиваем переменной i, установленное ранее, значение ШИМ на 5 выходе.
```

Указанных функций достаточно, для управления сервоприводами, или работой с Servo Shield как с расширителем выходов, но для реализации более сложных задач, в библиотеке реализованы дополнительные функции.

Описание дополнительных функций библиотеки:

Функция bus();

- Назначение: Устанавливает скорость работы Multi Servo Shield на шине I2C.
- Синтаксис: bus(ЧАСТОТА);
- Параметры:
  - ЧАСТОТА - указывается в кГц от 1 до ((F_CPU/10000)-16)/2 (по умолчанию 100 кГц)
- Возвращаемые значения: Нет.
- Примечание: Функцию можно вызывать как до, так и после функции begin.
- Пример:

```
MSS.bus(400);                          // Устанавливаем скорость работы шины I2C в значение 400 Кбит/сек.
```

Функция restart();

- Назначение: Перезагружает Multi Servo Shield, с установкой значений по умолчанию.
- Синтаксис: restart();
- Параметры: Нет.
- Возвращаемые значения: Нет.
- Примечание: Нет.
- Пример:

```
MSS.restart();                         // Перезагружаем Multi Servo Shield.
```

Функция invert();

- Назначение: Инвертирует сигналы на всех выходах Multi Servo Shield.

- Синтаксис: invert(ФЛАГ);
- Параметры:
  - ФЛАГ - принимает значение true или false, указывающее инвертировать или нет состояние на выходах.
- Возвращаемые значения: Нет.
- Примечание: Нет.
- Пример:

```
MSS.invert(true);                       // Инвертируем сигналы на всех выходах.
```

Функция outdrv();

- Назначение: Устанавливает схему с каскадным выходом выводов внутри чипа.
- Синтаксис: outdrv(ФЛАГ);
- Параметры:
  - ФЛАГ - принимает значение true (схема с каскадным выходом) или false (схема с открытым стоком).
- Возвращаемые значения: Нет.
- Примечание:
  - Схема с каскадным выходом используется если внешние устройства подключены к Shield через драйвер.
  - Схема с открытым стоком используется если внешние устройства подключены к Shield напрямую, без драйвера.
- Пример:

```
MSS.outdrv(true);                       // Устанавливаем подключение выходов чипа по внутренней схеме с каскадным выходом.
MSS.outdrv(false);                      // Устанавливаем подключение выходов чипа по внутренней схеме с открытым стоком.
```

Функция outState();

- Назначение: Устанавливает одно из трех состояний на всех выходах, если на входе OE установлена «1».
- Синтаксис: outState(СОСТОЯНИЕ);
- Параметры:
  - СОСТОЯНИЕ - одно из трех значений: LOW , HIGH или IMPEDANCE.

- Возвращаемые значения: Нет.
- Примечание: При указании HIGH и выборе схемы с открытым стоком, на выходах будет состояние IMPEDANCE.
- Пример:

```
MSS.outState(LOW);                          // При подаче 1 на вход чипа «OE», на всех выходах установится низкий уровень.
MSS.outState(HIGH);                         // При подаче 1 на вход чипа «OE», на всех выходах установится высокий уровень.
MSS.outState(IMPEDANCE);                    // При подаче 1 на вход чипа «OE», на всех выходах установится уровень высокого им
```

Функция extClock();

- Назначение: Устанавливает работу чипа от внешнего источника тактирования, с указанием его частоты.
- Синтаксис: extClock(ЧАСТОТА);
- Параметры:
  - ЧАСТОТА - число, от 1 до 50'000, указывает частоту внешнего источника тактирования в кГц.
- Возвращаемые значения: Нет.
- Примечание: Для возврата к внутреннему источнику тактирования, нужно указать параметр false.
- Пример:

```
MSS.extClock(10000);                    // Указываем чипу, работать от внешнего источника тактирования, настроенного на частот
MSS.extClock(false);                    // Указываем чипу, работать от внутреннего источника тактирования
```

Функция reg();

- Назначение: Чтение или запись одного байта данных в (из) регистр чипа.
- Синтаксис: reg(АДРЕС [, ДАННЫЕ]);
- Параметры:
  - АДРЕС - число указывающее адрес регистра, данные которого нужно прочитать/записать.
  - ДАННЫЕ - байт данных для записи.
- Возвращаемые значения: Прочтённые или записываемые данные.

- Примечание:
  - Если функция вызвана без параметра ДАННЫЕ, то она вернет байт данных находящийся в регистре с указанным адресом.
  - Если функция вызвана с параметром ДАННЫЕ, то она запишет указанные данные в регистр с указанным адресом.
- Пример:

```
byte i = MSS.reg(0x05);                 // Читаем байт данных из 5 регистра чипа в переменную i
MSS.reg(0x05,0x00);                     // Устанавливаем значение 0 в 5 регистр чипа
```

Применение:

- Управление сервоприводами (до 16 шт.)
- Устройство расширения выходов Arduino (не входов)